A METHOD TO PREVENT NET UPDATE OSCILLATION

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to bridged serial bus. More particularly, the present invention relates to net update oscillation problems in serial bus bridges such as the IEEE 1394.

2. Description of the Related Art

A net refers to a collection of buses which are interconnected by bridges. According to the IEEE definition, a bridge implements two portals and forwards asynchronous subactions, and may also forward isochronous subactions according to route information which is stored. Each bus that forms a net has an individual identifier that is unique if the net is stable and configured properly. In particular, in an IEEE 1394 net, there can be up to 1,023 logical buses and up to 63 nodes on each bus. Loop interconnection is physically allowed between buses within a net. However, loop(s), if any, shall be eliminated by muting at least one bridge per a loop according to IEEE1394 bridge draft standard. When two or more bridged nets are connected, a net update procedure is required.

According to IEEE1394.1 bridge draft standard version 1.00, prime portal is defined as a singular portal within a net of interconnected buses and clan is defined as a group of affiliated bridge portals that exhibit allegiance to the same prime portal. When two nets are connected, it is possible for bridge portals on the same bus to belong to different clans. However, this is a temporary condition that is resolved by net update procedures.

When a coordinator finds portals belong to different clans, the coordinator shall select one prime portal and update packet routing information according to IEEE1394.1 bridge draft standard. The coordinator, then, shall send an UPDATE_ROUTE message to each portal on its local bus in order to update each portal's packet routing information and clan affinity.

If a new update process is initiated while another net update(s) is(are) being processed, these net update processes shall be merged into one. Otherwise, a net can not be configured properly. According to the IEEE1394.1 draft standard version 1.00, only a coordinator can detect net update collision and initiate new update processes after selecting a surviving prime portal and updating packet routing information.

However, a problem arises in that if a bridge observes a net update process on one portal while processing another net update received on the other portal, two different UPDATE_ROUTE messages related to the two different net updates can be passed each other at the bridge without being merged into one. Then, both bridge portals are updated with the UPDATE_ROUTE messages exchanged each other, and then initiate bus resets on their local buses.

As a result, each coordinator on each bus observes a new different net update collision. Two coordinators then may send different UPDATE_ROUTE messages to each portal on its local bus because of the different net update collisions. If these happen, the same bridge will observe a net update process at one portal while processing another net update received at the other portal again. It could result in an infinite net update oscillation problem.

For example, Figure 1 illustrates a time flow diagram of an example that two UPDATE ROUTE messages are received by a bridge on its two local buses and the two UPDATE ROUTE messages will be passed by and be forwarded to the other buses.

A net update process starts when a bridge receives a net update message from a portal. Normally, after a coordinator (i.e. the bridge portal in charge of net updates) receives a net update message, the bridge's clan information is updated, and the process ends with the initiation of a bus reset on the other portal's bus.

The coordinator (not shown) on the first bus A (not shown) sends a net update message, which at point 120 is received by bus A. At approximately the same time 130, the coordinator (not shown) on bus B sends a net update to bus A. Thus there is a "crossing" of update messages between the start from one bus to the reset on the other bus, in the time frame referred to as a net update period (the period from 130 to 140).

When first bus A receives the clan information, which includes reset information from second bus B, it updates its clan information and a reset is initiated.

Thus, the respective coordinators for A and B will still detect "competition" regarding the bridge clan information and both will again perform net updates.

Subsequently, the two coordinators will initiate different update processes again. The bridge may then again receive two different net update messages on both portals by the crossing at 150. An infinite net update loop can be the result. In other words, the bridge continuously receives two net update messages on both portals and forwards them to the buses infinitely. In such a situation, the net configuration is never finished. This looping is referred to as net update oscillation.

SUMMARY OF THE INVENTION

According to an aspect of the present invention, a method to prevent net update oscillation includes the bridge selecting only one of UPDATE_ROUTE messages each of which is created by each portal itself if the portal is a coordinator on its local bus, otherwise sent by a portal's local coordinator and received by the portal. Then next, the bridge discards the other UPDATE_ROUTE message, so that an infinite net update oscillation cannot occur. The selected UPDATE_ROUTE message, which is referred to as a survived NET_UPDATE message, will be processed by the bridge according to IEEE 1394.1 bridge draft standard, therefore the portal that received the survived UPDATE_ROUTE message initiates a bus reset as an initiation of a net event, while the other portal doesn't. Then, a coordinator (which could be the portal) after the bus reset will find the net update process collision

and solve the issue according to IEEE 1394.1 bridge draft standard.

Since one of the net update events observed by the bridge has been discarded and only one net update event will be processed, net update events cannot be passed each other at a bridge. That can prevent the net oscillation problem explained above.

In a first aspect of the invention, a method for preventing net update oscillation comprises the steps of:

(a) determining whether a particular portal is a coordinator on its local bus; and proceeding to step (b) (i) if said particular portal is a coordinator, otherwise, proceeding to step (b) (ii);

(b) (i) determining whether said particular portal finds a net update collision on its local bus; and proceeding to step (c) if the net update collision is found;

(b)(ii) determining whether the particular portal receives an UPDATE_ROUTE message from another portal that is a coordinator on the local bus; and proceeding to step (c) if the UPDATE_ROUTE message is received;

(c) setting a global net_update bit to one by a lock procedure;

(d) verifying whether the lock procedure in step (c) has been successfully performed by determining whether the net_update bit has been set to one;

(e) performing one of:

(i) discarding the net update even if it has been determined in step (d) that the lock procedure in step (c) has not been successfully performed; and

(ii) processing the net update according to IEEE1394.1 bridge standard and setting the net_update bit to zero.

In another aspect of the invention, a method to prevent net update oscillation comprises:

(a) receiving on a first portal of the bus bridge a first net update message from a first coordinator on a first bus;

(b) receiving on a second portal of the bus bridge a second net update message from a second coordinator on a second bus before said first net update message has been processed by said first portal of said bus bridge;

(c) selecting and processing by the bridge of one of the first and second net update messages as a surviving net update message, and discarding the other of the first and second net update messages; and

(d) updating clan information so that both the first and second portal contains clan information of the surviving net update message.

Brief Description of the Drawings

Figure 1 is a time flow diagram showing how in the prior art infinite net oscillation can occur.

Figure 2 is a flowchart of a method of the present invention to prevent net update oscillation.

Detailed Description of the Invention

With regard to how the bridge selects the survived UPDATE_ROUTE message received by one portal and discards the other received, the following examples are for explanatory purposes only and do not limit the claimed invention to just these criteria for selecting a survived message and discarding a victim message.

(1) First in time; the bridge keeps the first received net update message on one bus and discards the second message;

(2) CPU priority, one CPU; the message found first by the CPU is kept and the other message is discarded; or

(3) CPU priority multiple CPUs; the first CPU that reports a net update event detection to the other CPU process it,

while the other CPU that detects subsequent net update events before the first net update event is completed will ignore the net update events; or

(4) Configuration of portals; one portal in a bridge can be designated as the survivor and the other as the victim. If each portal of the bridge receives an UPDATE_ROUTE message or founds a net update event then creates an UPDATE_ROUTE message as a coordinator, the NET_UPDATE message from the victim portal is discarded and the other NET_UPDATE message from the survivor portal is processed; or

(5) Using the same prime selection criteria specified by IEEE 1394.1 draft standard; but in this case the bridge may not edit any update messages. The bridge shall keep and process the UPDATE_MESSAGE corresponding to the survived prime selected according to IEEE1394.1 bridge draft standard, and discard the other UDPATE_ROUTE message.

Figure 2 shows a flowchart providing one example of ways of implementing the invention. A bridge implementing the procedure described by the following flowchart shall implement a global bit shared by the both portals. The global bit can be implemented in either software or hardware, and is called net_update bit for an explanatory purposes. A well-known lock procedure is used to set the global net_update bit to one. The global net_update bit can

be cleared to zero by a portal that has set it to one. The initial value of the net_update bit shall be zero.

At step 200, if the portal is a coordinator on its local bus, step 210 will be processed next, otherwise if the portal is not a coordinator on its local bus, step 420 will be executed next.

At step 210, if the portal finds a net update collision on the local bus according to IEEE1394.1 bridge draft standard, step 430 will be next performed. Otherwise, step 400 will be processed next.

At step 220, if the portal receives an UPDATE_ROUTE message from its coordinator on the local bus, step 230 will be performed next. Otherwise, the step 200 will be processed next.

At step 230, the global net_update bit is set to one by a well-known lock procedure. For example, at step 231, if the net_update bit identifies one, the lock fails. Otherwise, if the net update bit is set to zero, at step 232 the global net_update bit is set to one and the lock succeeds. If the lock of step 230 fails, step 260 will be next processed. Otherwise, the lock of step 230 successes, step 240 will be next processed.

At step 240, the net update is processed by the portal according to IEEE 1394.1 bridge draft standard. Step 250 will be next processed.

At step 250, the portal clears the net_update bit to zero.

At step 260, the portal discards the net update event. For this case, the other portal has been processing another net update found at the other portal's local bus.

As a result of step 260, one of two net update events found by the bridge is discarded and a possible net update oscillation problem can be prevented.

In all of the preceding an IEEE 1394 or equivalent thereof may be used, but the invention expressly is not limited to the IEEE 1394 and can be used on any serial bus bridge.